Routing Gateway and MCU Service Prefixes on the Cisco MCM Gatekeeper

Introduction

This paper provides guidelines for configuring the Cisco MCM Gatekeeper to properly route calls to MCU and gateway service prefixes in H.323 video conferencing networks. The goals of this paper are:

1. Introduce the overall design model of an IP/VC videoconferencing network.
2. Discuss in detail the parsing logic used by the Cisco MCM Gatekeeper when performing call routing and address resolution. It is very important to understand this logic, so as to properly design a dial-plan that is consistent with the gatekeeper's method of parsing the dialed number. Designing a successful dial-plan also consists of ensuring that numbers used for different purposes do not overlap (i.e. zone prefixes, service prefixes and E.164 addresses must be unique).
3. Discuss in detail how the Cisco IP/VC MCU and Gateways register their service prefixes with the Cisco MCM Gatekeeper.
4. Discuss interoperability issues with other vendor's MCU and gateway products.

Where appropriate, this paper refers to existing procedures in the following Cisco documents:

Cisco IOS Software Configuration
Cisco Multimedia Conference Manager Configuration Guide
Cisco IP/VC Videoconferencing Design Guide
Cisco IP/VC 3510 MCU User Guide
Cisco IP/VC 3520 Gateway User Guide
Cisco IP/VC 3525 Gateway User Guide

The procedures in this paper assume familiarity with Cisco IOS Software. For instructions on using Cisco IOS Software, please refer to the Cisco IOS Software Documentation Set referenced above. In addition, this paper assumes some level of familiarity with the use and capabilities of the Cisco IP/VC product family, and the Cisco Multimedia Conference Manager.

Note: *For those who are new to the Cisco IP/VC videoconferencing product family and the Cisco MCM Gatekeeper, it is highly recommended that you first review the users guides referenced above, as this paper is designed to enhance your understanding of the products beyond that of the new user, and help you understand how to properly design and configure larger, multi-zone H.323 videoconferencing networks.*

Copyright © 2001 Cisco Systems, Inc.

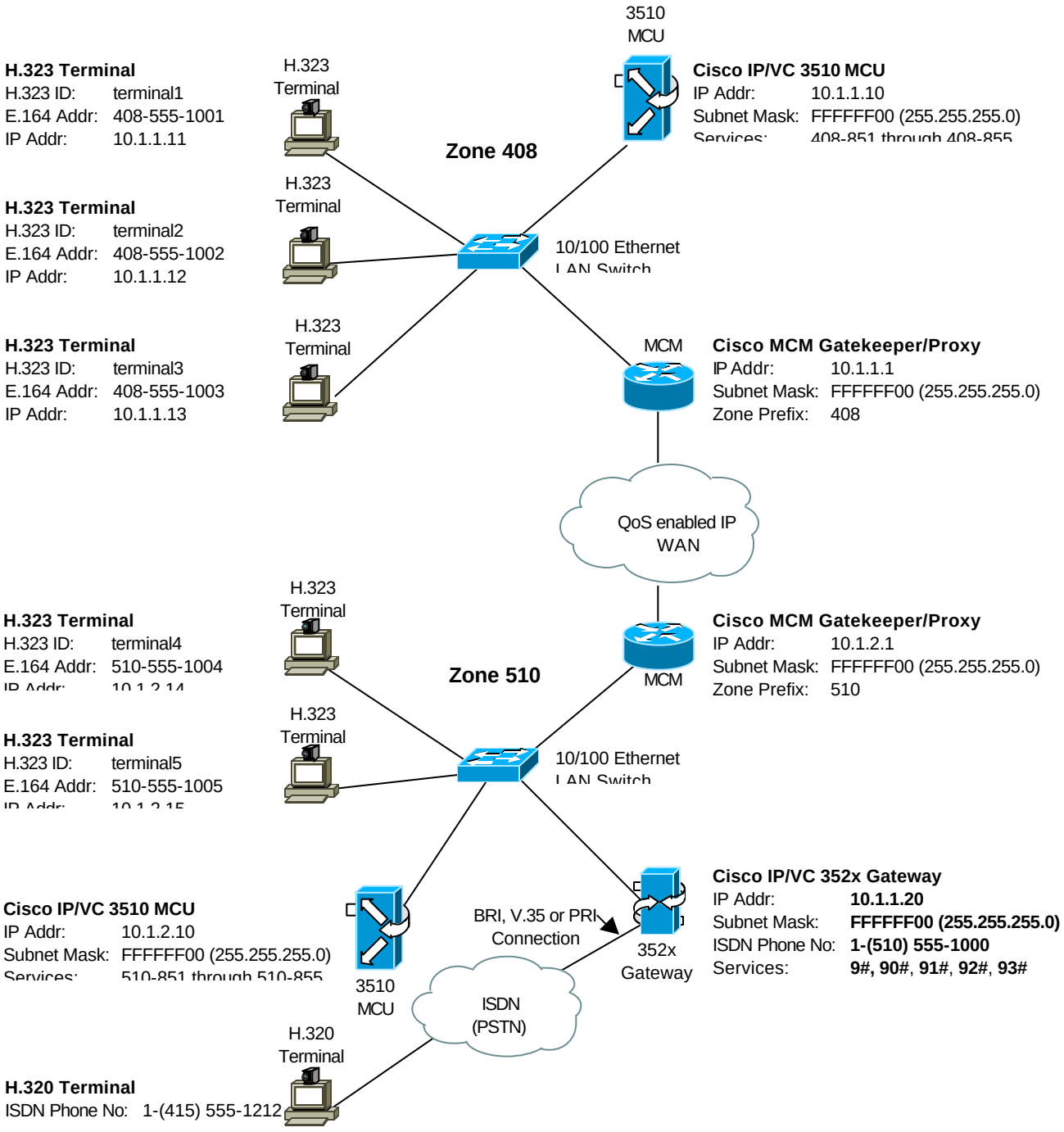
Kevin McMenamy SH1 - 1/24/01

Table of contents

Introduction 1
Table of contents 2
Example of a Cisco IP/VC H.323 network 3
Table of service prefixes depicted in figure 1. 4
Choosing your Service Prefix Digits Carefully 6
How the Cisco MCM Gatekeeper views service prefixes differently than it views E.164 addresses 6
The parsing order the Cisco MCM Gatekeeper uses when processing admission requests and location requests (ARQ's/ LRQ's) 7
Why service prefixes must not conflict with any known zone prefixes or E.164 addresses 9
Why is it important that you ensure the service prefixes you choose enable "user-friendly" dialing strings (i.e. 4-digit, 7-digit, 10-digit dialing) 10
Service prefixes and interzone call routing. 11
How to design your service prefixes to ensure reachability from endpoints in any zone 11
When should you use the Cisco MCM Gatekeeper "hopoff" commands versus routing calls based on zone prefixes 12
Routing service prefixes in hierarchial network topologies 16
Registering your service prefixes with the Cisco MCM Gatekeeper 19
How the IP/VC 3510 MCU registers its service prefixes with the Cisco MCM Gatekeeper, and how it differs between version 1.x and 2.x 20
How the IP/VC 352x Gateways register their service prefixes with the Cisco MCM Gatekeeper, and how it differs between versions 1.x and 2.x 22
How other vendor's MCU's and gateways work with the Cisco MCM Gatekeeper. 23

Example of a Cisco IP/VC H.323 network

Figure 1 **Example IP/VC network topology**



The IP/VC network depicted in figure 1 is the model for the configuration procedures in this paper, and illustrates a sample H.323 network, in which there are two H.323 zones managed by Cisco MCM Gatekeepers.

In each of the zones, there is a Cisco IP/VC 3510 MCU which have each registered several service prefixes with their respective gatekeepers; prefix numbers 408851 through 408855, and 510851 through 510855 respectively.

In the 510 zone, there is also a Cisco 352x Gateway which has registered four service prefixes with its gatekeeper; prefix numbers 9#, 90#, 91# and so on. *Did you notice that there is no gateway in the 408 zone? This means that the terminals in the 408 zone will need to have access to the remote gateway in the 510 zone.*

Finally, each H.323 terminal has been assigned a unique E.164 address, which matches the zone prefix that the terminal is registered in.

Table of service prefixes depicted in figure 1.

The following table explains the service prefix numbers, and their related parameters, for the MCU and gateway depicted in figure 1.

Table 1 MCU and gateway service definitions

MCU Service Prefix	Service Parameters	Gateway Service Prefix	Service Parameters
408851 = 510851 =	15 users @ 128kbps, voice-activated switching	9# =	64kbps (voice only)
408852 = 510852 =	6 users @ 128kbps, continuous presence	90# =	2x64kbps
408853 = 510853 =	9 users @ 384kbps, voice-activated switching	91# =	128kbps
408854 = 510854 =	4 users @ 384kbps, continuous presence	92# =	384kbps
408855 = 510855 =	5 users @ 768kbps, voice-activated switching	93# =	768kbps

These prefixes are simply given as an example of how services may be configured for your network. The parameters of each service are not the focus of this paper. Rather, we will simply focus on how to route those prefixes through the gatekeeper.

After examining the network illustrated in figure 1, you may have several questions, such as,

- "Why did you decide to use zone prefixes that are similar to PSTN area codes?"
- "Why do the MCU service prefixes begin with 408 or 510, while the gateway service prefixes do not begin with 510?"
- "What made you decide to use 10-digit numbers for the H.323 terminals? Couldn't you just use 4-digit extensions instead?"

- “Do the users have to dial the full number, starting with the zone prefix, even if they are in the same zone as the destination? Or can they simply dial the extension of the destination, leaving off the leading zone prefix digits?”
- “What made you decide to use numbers starting with “8” for MCU service prefixes, numbers starting with “9” for gateway service prefixes, and numbers starting with “555” for endpoints; and why do you have the “#” sign after each of the gateway service prefixes?”

The purpose of this document is to help you understand the answers to these and other questions. There are valid technical reasons for these design criteria. If you do not configure your service prefixes properly, you will be unable to route calls from one zone to another, or worse, you could end up with conflicting numbers, resulting in calls being mis-directed to the wrong destination!

The configuration illustrated in figure 1 is not the ONLY way one can configure these service prefixes and E.164 addresses, but it provides an example of a consistent, effective solution without conflicts or limitations. This paper is designed to explain how call routing works, and once you understand that, you will be able to choose how to best configure your dial plan to work in each customer’s environment.

Let’s start with a detailed look at how the Cisco MCM Gatekeeper routes calls, and why it is so important to design your network routing patterns and dial plan appropriately.

Choosing your Service Prefix Digits Carefully

This section provides an overview of service prefixes, as they relate to the gatekeeper's call routing functions. In addition, it provides an overview of how to choose which digits, or ranges of digits, to use for service prefixes throughout your network.
This section will cover the following topics.

- How the Cisco MCM Gatekeeper views service prefixes differently than it views E.164 addresses.
- The parsing order the Cisco MCM Gatekeeper uses when processing admission requests and location requests (ARQ's/ LRQ's).
- Why service prefixes cannot conflict with any known zone prefixes or E.164 addresses.
- Why it is important that you ensure the service prefixes you choose are useable, and enable "user-friendly" dialing strings (i.e. 4-digit, 7-digit, 10-digit dialing).

How the Cisco MCM Gatekeeper views service prefixes differently than it views E.164 addresses

In the H.323 standard, there are two mechanisms for associating a numeric value to a device; E.164 addresses and service prefixes (also referred to as technology prefixes). E.164 addresses are typically used for terminals, or any device that is capable of handling a single call at a time. Service prefixes are used to address devices which offer a "service" to other terminals, and which are therefore capable of handling multiple simultaneous calls. It is important to understand how the Cisco MCM Gatekeeper treats these two types of numeric values, when deciding how to route a call based on the number that was dialed.

There is a fundamental difference in the way the Cisco MCM Gatekeeper views service prefixes compared to the way in which it views E.164 addresses.

The MCM Gatekeeper's Use of Service Prefixes

The gatekeeper uses a logical wildcard mask (*) at the end of a service prefix: for example, 408853*. The asterisk means that 408853 and "any digits following", would be routed to the provider of that service, in this case the MCU in the 408 zone (see figure 1).

Note: *It is important to note that the user does not have to actually dial the * character! Likewise, the administrator who configures the service prefixes on the MCU and gateway does not have to put the * at the end of the prefix. The only place the asterisk is actually used is in the gatekeeper's zone definition tables. Examples of these tables will be shown in a later section.*

The asterisk is a logical representation in the gatekeeper to denote, "*and any digits following the number before the asterisk*". The use of the asterisk informs the gatekeeper that any number that begins with 408853 should be routed to the device that has registered that service prefix. This allows additional digits to be specified after the service prefix.

For example, if Terminal1 wanted to dial into a conference on the MCU, the user could dial 4088531212, where "408853" is the service prefix, and "1212" is the unique conference ID to be created or joined.

Likewise, in the case of a Gateway call from Terminal4 to the H.320 Terminal, the user could dial 9#14155551212, where "9#" is the service prefix, and "14155551212" is the PSTN number that the gateway should place the call to. From the gatekeeper's perspective, it does not even look at the numbers following the service prefix. It sees the service prefix 9#*, and matches the dialed number

"9#14155551212". The asterisk is a wildcard to match the "14155551212" portion of the dialed number.

The MCM Gatekeeper's Use of E.164 Addresses

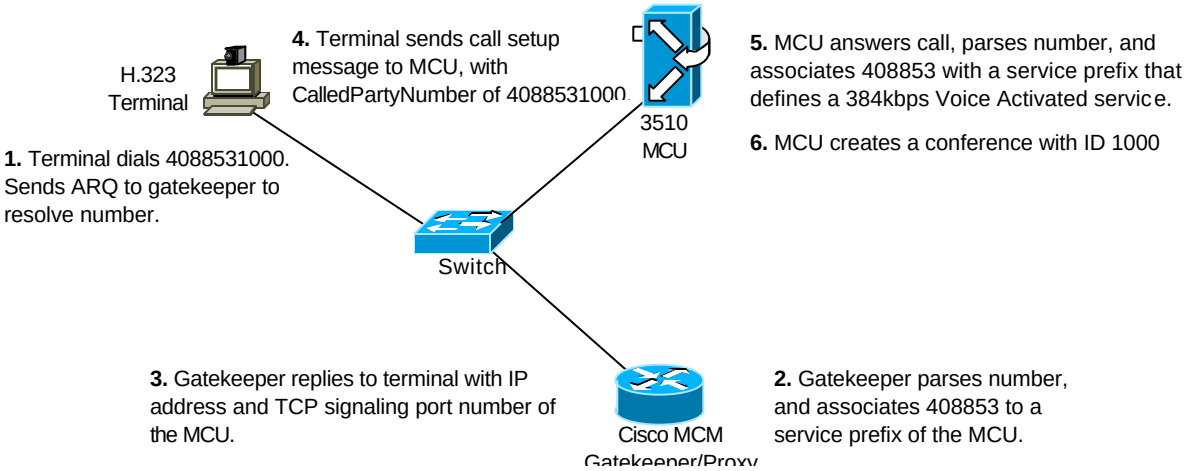
An E.164 address, on the other hand, is viewed by the gatekeeper in its entirety. If the service prefix 408853 were registered as an E.164 address, and the caller placed a call to an MCU conference, such as 4088531234 (with 1234 being the conference ID), the gatekeeper would not find a match for this destination in its E.164 registration table, since the only digits registered would be 408853. Therefore, the call would fail, with a causecode "CalledPartyNotRegistered" being returned by the gatekeeper.

This is why service prefixes are used, to enable the use of additional digits being specified in the dial string after the service prefix. The use of service prefixes allows the caller to reach the gateway or MCU by way of a prefix, and then denote the additional digits to be used for the completion of the call. The gatekeeper only looks at the service prefix, and ignores any digits following that prefix.

Example Call Flow from a Terminal to an MCU

Figure 2 illustrates a call placed from Terminal1 to an MCU conference using service prefix 408853.

Figure 2 Example call flow from a terminal to an MCU



This shows that the use of a service prefix allows a terminal to access an MCU or a gateway by dialing its prefix, and then following the prefix with additional digits, such as a conference ID in the case of an MCU call above. This is possible because when the gatekeeper parses the dialed number, it does not look for an "exact match" for 4088531000, but rather finds a match on 408853*.

We'll now look at the order in which the gatekeeper resolves these different types of numbers.

The parsing order the Cisco MCM Gatekeeper uses when processing admission requests and location requests (ARQ's/ LRQ's)

It is very important to understand the parsing order the gatekeeper uses when processing call requests, as it attempts to match the called number to the approriate destination. The gatekeeper must be capable of detecting whether the call is destined for an E.164 address registered to a terminal in its local zone; a service provider, such as a gateway or MCU; or a remote zone prefix.

The parsing order is as follows on the Cisco MCM Gatekeeper:

1. Service prefix (also referred to as a technology prefix)
2. Zone prefix
3. E.164 address

Here are some sample call scenarios, again referring to the example network shown in figure 1.

Example 1: A call from Terminal 4, to Terminal 1 in the 510 zone.

To reach Terminal1 in the 510 zone, Terminal4 in the 408 zone would dial <zone prefix> <E.164 address>. For example:

4085551001

In this case, when the gatekeeper in the 408 zone recieves the call request, it will first attempt to match against any registered service prefixes. The gatekeeper knows about service prefixes 408851* through 408855*, but it does not have a service prefix entry for a number starting with 408555*. It then continues to compare the dialed number to all known zone prefixes. In this case, the gatekeeper finds a match in its zone prefix database for 408* and determines that this zone prefix is its own local prefix. Therefore, it continues to parse the remaining digits of the dialed number against all known E.164 address entries in its local Endpoint Registration Table. It finds a match for 4085551111 on Terminal1, and returns the IP address and TCP signalling port information of Terminal1 to the originator of the call. Terminal4 then establishes a call with Terminal1.

Example 2: A call from Terminal4, to an MCU service in zone 408, such as 408853.

To reach the MCU Service 408853, Terminal4 would dial <zone prefix> <MCU Service prefix> <Conference ID>. For example:

4088531234

In this case, Terminal4 wants to join a conference being hosted on the MCU of 4088531234, with 1234 being the conference ID. Terminal4 initiates the call by dialing 4088531234. When the gatekeeper in the 408 zone receives the call request, it performs the same parsing order as described above. However, in this case it immediately finds a match for service prefix 408853* on the MCU, and returns the IP address and TCP signalling port information of the MCU to the originator. Terminal4 then establishes a call with the MCU, and the MCU adds Terminal4 to the 4088531234 conference.

Now that you understand the parsing order, consider what would happen if the MCU did not register its service prefixes with the digits “408” in front of each one, but simply registered its service prefixes as 851, 852, 853 and so on. *What would be the affect on the completion of the call, given the gatekeeper’s parsing order?*

Let’s take the same call example as #2 above, where Terminal4 dials 4088531234. However, assume that the MCU has only registered 853* as a service prefix with the gatekeeper instead of 408853*. In this case, when the gatekeeper in the 408 zone receives the call request, it will first attempt to match the dialed number against any known service prefixes. Not finding a service prefix for 408853*, it will then attempt to match the dialed number against all known zone prefixes. In this case, the gatekeeper finds a match in its zone prefix database for 408* and determines that this zone prefix is its own local prefix. Therefore, it then continues to parse the remaining digits of the dialed number against all known E.164 addresses in its local Endpoint Registration Table. It will fail to find an E.164 address of 4088531234. It knows about the service prefix 853*, but does not find an exact match for 8531234 in its E.164 address table. Therefore, the call will fail!

You may think, “*but the number DOES match the service prefix!, why can’t the gatekeeper go back up and find the 853* service prefix after it has determined the 408 to match its local zone prefix*”?It is because this is the way that the Cisco MCM Gatekeeper parsing order works.

- If it does not match the service prefix the first time down the list, it will not go back up and look for a matching serving prefix again after matching on the zone prefix.
- It will only go down and try to match against all local E.164 addresses.

So because the Cisco MCM Gatekeeper parses the dial string against service prefixes FIRST, and THEN against the zone prefix table, it is imperative that you place the zone prefix in front of each service prefix on the MCU to ensure successful call routing`.

Note: *The alternative to doing this is to use the Gatekeepers "hopoff" command instead. See the section on hopoffs later in this paper for more details.*

Why service prefixes must not conflict with any known zone prefixes or E.164 addresses

Since the Cisco MCM Gatekeeper parses the dial string in the order of (1) service prefixes, (2) zone prefixes and (3) E.164 addresses, it should be apparent why a service prefix must not conflict with any zone prefixes associated with remote gatekeepers, or E.164 addresses registered to your terminals.

Zone Prefix Conflicts

Consider why service prefixes cannot conflict with any remote zone prefixes. Referring back to figure 1 above, you will recall that the gateway in figure 1 has a service prefix starting with the digits 91. Now, assume that we added a third zone to the network, and assigned it a zone prefix of 916* If we did not append the # character after the service prefix of the gateway, and a user in the 408 zone dialed 9165551212, what would be the result?

- Recall again the parsing order of the Cisco MCM Gatekeeper, which is (1) service prefix, (2) zone prefix and (3) E.164 address, and you will see that the call would be routed to the gateway, and not to the 916 zone!
- The call would arrive at the gateway, and the gateway would attempt to dial the digits following the service prefix, which in this case would be 65551212.
- This call would obviously fail in the PSTN! (unless of course there was a valid number in the local area code of 655-5121).

For this reason, it is crucial that you configure your zone prefixes and service prefixes to eliminate the potential for conflicts. There are two useful options for doing this:

1. Append a "#" after each of your gateway service prefixes. Using this option, you do not need to modify your zone prefixes, since the use of the # character sufficiently differentiates it from any possible zone prefixes.
2. Prepend a "1" in front of your zone prefixes. For example, using this method, the zones in figure 1 would change from 408, 510 and 916 to 1408, 1510 and 1916 respectively.

Note: *In figure 1, we have chosen to illustrate the first option. However, the second option would be just as efficient in eliminating any potential conflicts between the 91 service prefix and 916 zone prefix.*

E.164 Address Conflicts

Now consider why service prefixes cannot conflict with any E.164 addresses. For example, assume that Terminal1 registered with the E.164 address 4088531111 (instead of 4085551111 as shown in figure 1). If Terminal4 attempted to dial Terminal1 by dialing 4088531111, what would be the result?

- The gatekeeper in the 408 zone will attempt to first match the dialed number against any known service prefixes.
- It will match the dial string to service prefix 408853*, and route the call to the MCU in the 408 zone, rather than to Terminal1.

This is why E.164 addresses and service prefixes must not conflict with each other, and why it is usually recommended that you choose a digit range for MCU and gateway service prefixes that is not likely to be used by any endpoints.

For example, you could use the range of numbers from 408850 through 408859 for MCU service prefixes, and ensure that other digit ranges are available for endpoints to use as E.164 addresses, such as 408860*xxxx* through 408899*xxxx*, etc.

This is just one example of digit ranges that could be used. Ultimately is it entirely up to the network administrator as to which ranges will be used for what purpose.

Why is it important that you ensure the service prefixes you choose enable "user-friendly" dialing strings (i.e. 4-digit, 7-digit, 10-digit dialing)

At this point, you have learned how service prefixes work from a call routing perspective, and why you should give careful consideration to what digits or digit ranges are used for service prefixes. In the following paragraphs, we will exand the discussion to help you contemplate another important aspect that must be considered when choosing service prefixes which is how and why to choose digit ranges that are intuitive for users to read and use.

You must consider whether the numbers you implement will be intuitive and easy for users to understand and remember. Forcing an average user to memorize and dial long strings of digits is in most cases unacceptable. Since most users are familiar and comfortable with the dialing patterns used with telephones, you could choose to implement your service prefixes so that they simulate a PSTN number, such as a seven-digit or ten-digit number. In other cases, you may choose to implement a number scheme that would emulate an *n*-digit extension type of dial plan, where users simply dial the extension of the destination device, which could be four, five or more digits.

Regardless of what you ultimately choose to deploy for a number scheme, you will have to configure your service prefixes to match your dialing plan scheme. In the example given in figure 1, we show the MCU configured with a service prefix of 408853*. This allows for add-hoc conferences to be established by dialing:

- 408, which is the zone prefix
- 853, which is the service prefix that denotes a particular type and speed of conference
- A four-digit conference ID, which establishes a unique conference for a particular group of users.

So, for example, if terminals 1 through 4 all wanted to participate in a 384kbps conference on the MCU in the 408 zone, they would each dial;

4088531234

This dial string would cause the MCU to establish a conference using the 408853 service, with a conference ID of 1234.

By using a zone prefix that resembles an area code, another three digits for the service prefix, and finally a four-digit conference ID, this call looks exactly like a standard ten-digit number that one might dial in the PSTN telephony world. This scheme allows for a user-friendly dialing approach as users become familiar with H.323 videoconferencing.

Note: *This subject is being stressed because the Cisco IP/VC MCU's and Gateways ship with default service prefixes that DO NOT match this logic. The MCU's ship with services prefixes xx* through xx*, and the gateways ship with service prefixes xx* through xx*. These numbers are fine for small, single-zone networks. But for larger, multi-zone, multi-regional network designs, you will most likely choose to modify those numbers to something that more closely resembles a PSTN-like dialing experience for your users.*

Service prefixes and interzone call routing

This section deals with interzone call routing of service prefixes. Depending upon your network topology, you may need to ensure multi-zone, and even hierarchial-zone topologies, which adds complexity to your overall network dialplan and your use of service prefixes. This section will cover the following topics.

- How to design your service prefixes to ensure reachability from endpoints in any zone.
- When you should use the Cisco MCM Gatekeeper "hopoff" statements versus routing calls based on zone prefixes.

How to design your service prefixes to ensure reachability from endpoints in any zone

Given the network topology example shown in figure 1, let us consider the importance of how to ensure reachability to service prefixes from any zone on the network. In figure 1, we have a Cisco 3510 MCU in the 408 zone, and one in the 510 zone. In addition, the 510 zone has a Cisco 352x Gateway for access to the PSTN. However, there is no gateway in the 408 zone. Therefore, endpoints in the 408 zone that wish to access H.320 endpoints on the ISDN/PSTN network, must access the PSTN through the gateway in the 510 zone. To enable access between zones, you must consider the service prefixes used by the MCU's on the network.

Using Common Service Prefixes in All Zones

To ensure a dial plan that is consistent and user-friendly, we have chosen to use the same base service prefixes on both MCU's. However, we have pre-pended the zone prefix that each MCU resides in, in front of the base prefix, in order to form each complete service prefix. By doing this, users in the 510 zone who wish to access the MCU in the 408 zone would simply dial 408 and the service prefix they wish to use. Dialing the zone prefix and the service prefix for an MCU call in a remote zone enables a user-friendly dial format. However, if that same user in the 510 zone wanted to establish a conference using the MCU in the same zone, zone 510, he or she would still have to dial the zone prefix, rather than simply dialing the 85*x* service prefix. This may seem counter-intuitive, but there is an important reason for doing this.

- If the MCU's did not prefix their services with the zone prefix, there would be no way to route from a remote zone to that MCU. This is because of the parsing order that the Cisco MCM Gatekeeper uses when processing inbound call requests, as discussed above.
- If the MCU registered with a service prefix of 853, when the call is received by the gatekeeper, it will first attempt to match the dialed number against all known service prefixes in its zone. In this case, the dial string would be 4088531234.
- The gatekeeper would not find a service prefix beginning with 408853 *(even though the 853 service prefix does exist)*, and would then continue to parse the number against any known zone prefixes.

- It would find a match in its zone prefix database for 408* and determine that this zone prefix is its own local prefix.
- It would then continue to parse the number to find an endpoint with the E.164 address of 8531234, and would not find one. Therefore, the call would fail. This is why we must prefix the MCU service with the zone prefix, in this case 408853.

We recommend this method so that all MCU's across the network can all use the same service prefix numbers to denote the same settings. In other words, you could configure all of your MCU's to offer the same set of services *(see Table 1 for an example, where both MCU's offer identical services)*. This allows the user to only have to remember what service prefix offers the desired type of service *(i.e. Voice Activated Switching or Continuous Presence, speed, resolution, frame-rate, number of parties allowed, etc)*. The user then simply needs to know the zone prefix of the appropriate MCU to call into for that conference *(i.e. 408 or 510 in our example)*.

Using Unique Service Prefixes on all MCUs

The alternative to this would be to configure unique service prefix numbers on all MCU's. For example, you could use 60* through 65* for the MCU in the 408 zone, and 66* through 69* for the MCU in the 510 zone.

This would allow the users to call into the MCU without having to dial the zone prefix every time. However, the user would now have to memorize which service prefixes goes to which MCU. We believe this is more counter-intuitive that the previous method of using zone prefixes in front of the MCU service prefixes. Nevertheless, some customers may decide that this is how they want to do it.

Either way is fine, but the latter method introduces issues with routing the service prefixes in a multi-zone network. In order to route them, you are forced to use "hopoff" commands in all of your gatekeepers, so that the gatekeeper in the 510 zone would know that 6x* is to be routed to the 408 zone. The next section discusses the use of hopoff commands, and the problems associated with using them.

When should you use the Cisco MCM Gatekeeper "hopoff" commands versus routing calls based on zone prefixes

The hopoff command is used to statically program the gatekeeper to know how to route service prefixes that are not pre-pended by a zone prefix. For example, referring back to the topology illustrated in figure 1, instead of pre-pending the MCU services with 408, we could have instead programmed the 510 gatekeeper to know about service prefix 853, and tell it to forward all requests to that number to the 408 zone, by way of a hopoff command.

The hopoff command in the Cisco MCM Gatekeeper looks like this:

gw-type-prefix *prefix* **hopoff** *remote-zone-gatekeeper-name*

For example:

`gw-type-prefix` `853*` **`hopoff`** `gk-408`".

By doing this, you eliminate the need for the user to specify a zone prefix in front of the service prefix when dialing that service. ***However, by doing this, you also are no longer able to use the same service prefixes in all MCU's in all zones.***

Note: *The abbreviation "gw" is used to denote both gateway services and MCU service prefixes. Do not be confused by this. The Cisco MCM Gatekeeper currently treats MCU service prefixes and gateway service prefixes the same, and refers to them all collectively as "gw" service prefixes. In the*

future, a new command on the MCM may be introduced to differentiate between the two (i.e. "mcu-type-prefix <prefix> …"

For example, assume that the gatekeeper in the 510 zone has such a hopoff command to the MCU service prefix "853" in the 408 zone. How would this affect the use of the service prefix 853 on the MCU in the 510 zone?

- If Terminal4 in the 510 zone wanted to establish a conference using the local MCU in the 510 zone, and dialed 8531234, because the gatekeeper in the 510 zone has a hopoff command for 853 pointing to the 408 zone, it would send the call request to the 408 zone every time, and would never send the call to the local MCU, even though the local MCU is registered with the same service prefix!
- This is because hopoff statements override the parsing order described previously. If a hopoff statement is found, the gatekeeper does not continue to parse the dial string against any zone prefixes, or locally registered devices, but immediately forwards the call to the zone defined in the hopoff statement.

Arguments against using hopoffs

Because of this behavior, if you were to use hopoff statements for all your service prefixes, instead of using zone prefixes in front of all your service prefixes, it would require that **every** MCU, in **every** zone, be configured with "globally unique" service prefixes.

For example, the MCU in the 408 zone could use 850* through 855*, while the MCU in the 510 zone could use different numbers, such as 860* through 865*. In a two zone network, such as the example given in figure 1, this is not a big problem. But consider the use of globally unique service prefixes in a network of 100 zones!

- Being forced to use globally unique service prefixes for all your MCU's would use up a lot of number ranges, which would no longer be available for use as zone prefixes or as E.164 addresses.
- In addition, the users would have to keep a list of which service prefixes go to different MCU's. It would be very counter-intuitive for users to have to memorize this type of information!

For this reason, even though using the zone prefix in front of the service prefix requires that users always dial the zone prefix in front of the MCU service prefix, it is better in most cases than using hopoff statements for MCU service prefixes.

Another reason why the use of hopoff statements is limiting is that these hopoff statements would have to be manually programmed into EVERY gatekeeper in the network, for EVERY service prefix in the network. As the number of MCU's and service prefixes in the network grow, this could be a laborious task to statically program each gatekeeper with a list of hopoffs to every service prefix. *(You could use the Directory Gatekeeper feature to ease this administrative burden, but it would still be labor intensive, and counter-intuitive for your users. The use of the Directory Gatekeeper feature is explained below, in the section entitled, "Routing service prefixes in hierarchical network topologies").*

This would be equivalent to building a large routed IP network without the use of a dynamic routing protocol. You would have to manually enter static routes for every possible destination network. It is far easier from an administrative perspective to use a service prefix that is front-ended with the zone prefix, and allow the gatekeepers to route the calls naturally using their standard parsing logic and zone routing tables.

Where hopoffs ARE useful

However, having explained the use of hopoff statements, and the fact that they are not recommended for MCU services, the one place where hopoffs ARE useful, is for gateway service prefixes in zones that do NOT contain a gateway. A hopoff statement can be added to the gatekeeper to point calls to a zone that DOES offer gateway services.

Rather than pre-pending your gateway service prefixes with the zone prefix that the gateway resides in, as is recommended for MCU's, it is better to configure hopoff statements to those gateway services, if the gateway is not in the zone in which the call is originating from. This is because it is counter-intuitive for a user to have to dial a zone prefix, then a gateway service prefix *(such as 9# in the example shown in figure 1)*, then the PSTN number that they are trying to call.

For example, consider the following call scenario.

Example 1: A call from Terminal1, in the 408 zone, to the H.320 terminal in the (415) area code WITHOUT the use of a hopoff statement.

To reach the H.320 terminal, Terminal1 would dial <zone prefix><service prefix><1><area code><PSTN number>. For example:

5109#14155551212

Without using a hopoff statement in the 408 gatekeeper for service prefix 9#, the user is forced to dial the zone prefix, then the service prefix, then the PSTN destination number. Since the zone prefix in this case mimicks a PSTN area code, the user must dial an "area code" TWICE! This creates a very counter-intuitive dialing plan, and forces the user to understand what zone the gateway is in, and how to properly dial a digit sequence that will be successfully routed by the gatekeepers and the gateway.

Instead, it would be simpler for the user to simply dial the gateway service prefix (i.e. 9#), then the PSTN number they are calling, regardless of what zone they are in. Users are accustomed to dialing a prefix to reach an outside line. On most corporate phone systems, you must dial a "9" to reach an outside line. In our case, we have simply added a "#" character after the 9 to denote a service prefix on a videoconferencing gateway *(the use of the # character is described above in the section entitled, "Why service prefixes cannot conflict with any known zone prefixes or E.164 addresses")*.

Therefore, for zones that do not contain a local gateway, a hopoff statement can be added to the gatekeeper to point calls that begin with the digit 9 to a zone that DOES offer gateway services.

Example 2: A call from Terminal1, in the 408 zone, to the H.320 terminal in the (415) area code WITH the use of a hopoff statement.

To reach the H.320 terminal, Terminal1 would dial <service prefix><1><area code><PSTN number>. For example:

9#14155551212

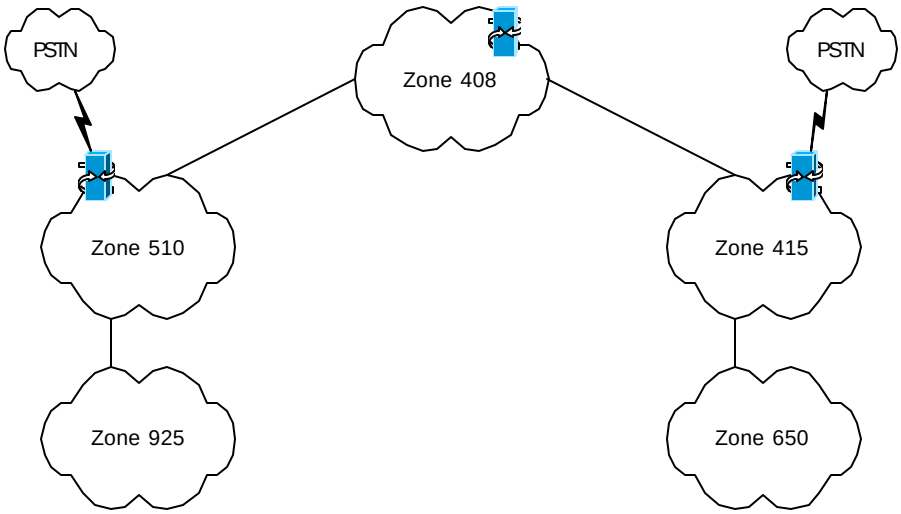
By comparing the two examples above, you can see that the only difference is that the hopoff statement relieves the user of the need to dial the zone prefix of the gateway. This makes it easy for all users within the organization to use gateway services. The users in the 510 zone would dial a gateway call just like the users in the 408 zone. The gatekeeper in the 510 zone would have the gateway services 9# through 93# registered locally, and the gatekeeper in the 408 zone would have a hopoff statement to 9* pointing to the 510 gatekeeper. The hopoff statement in the 408 zone gatekeeper would look like this:

"gw-type-prefix *9** **hopoff** *gk-510***"**

This method is recommended because it is both user friendly and highly scalable, even in large multi-zone, and even hierarchial-zone networks. Consider the following network example of a larger multi-zone, hierarchial topology.

Example of a larger multi-zone, hierarchial H.323 network

Figure 3 Example of a larger multi-zone, hierarchical H.323 network



The network in figure 3 illustrates the concept of a larger, hierarchial network topology, where the 408 zone is the main company headquarters, and there are two regional offices. Each regional office also has a “stub” zone reporting to it.

Since the stub zones (zones 925 and 650) only have a few users in them, the company decided NOT to purchase a gateway for those locations, but to allow those users access to the PSTN through the gateways in their respective parent zones.

In order to achieve a user-friendly dialing plan, each of the gateways offer the identical service prefixes. Each of them offers the services 9#, 90#, 91#, 92# and 93#. This way, no matter what location, building, or conference room the user may be in, he or she knows that dialing 9# will access a gateway service at 384kbps. The user doesn’t need to know, or care, where the gateways physically are! They simply dial 9#, and the PSTN number they are calling, and the call goes out a gateway “somewhere”…

In order to accomplish this, the best method would be to simply place hopoff statements in the 925 and 650 gatekeepers. The 925 gatekeeper would have a hopoff command to the gateway in zone 510, and the 650 gatekeeper would have a hopoff command to the gateway in the 415 zone.

Example 1: gk-925 hopoff command

```
gw-type-prefix 9* hopoff gk-510
```

Example 2: gk-650 hopoff command

```
gw-type-prefix 9* hopoff gk-415
```

For the users in all the other zones, dialing 9# would send them to their local gateway in their zone. For example, if a user in the 408 zone dialed 9#, the call would go out the gateway in the 408 zone.

If a user in the 510 zone dialed 9#, the call would go out the gateway in the 510 zone. But if a user in the 650 zone dialed 9#, the go would get forwarded to the 415 gatekeeper, and the call would go out the gateway in the 415 zone.

Routing service prefixes in hierarchial network topologies

In order to ease the administrative burden of programming gatekeepers to route service prefixes and zone prefixes, the Cisco MCM Gatekeeper has a unique feature known as “LRQ Forwarding” (also commonly referred to as the “Directory Gatekeeper” feature). This feature allows you to configure the lower-level gatekeepers in the hierarchy with minimal information, and to rely on the next higher-level gatekeeper in the hierarchy to resolve calls that the lower-level gatekeeper does not know how to handle.

Several excellent whitepapers have been written on the use of the Directory Gatekeeper feature, so we will not deal with it in its entirety. We will only mention it in enough detail as is relevant to the discussion and examples in this paper. One excellent place for additional information on Directory Gatekeeper can be found at http://von.cisco.com/Solutions/solutions/directory-gk.htm

In the example network scenario depicted in figure 3, the Directory Gatekeeper could be used very effectively to minimize the configuration needed in the regional and stub gatekeepers (zones 510 and 925, and 415 and 650). In this case, the gatekeeper in the 408 could become the master “Directory Gatekeeper”. This gatekeeper would know about all zone prefixes for every gatekeeper in the network. The configuration of the 408 gatekeeper would look like this.

Figure 4 Configuration for gk-408

```
gk-408#show run
. . .snipped for brevity. . .
gatekeeper
zone local gk-408 cisco.com 1.1.1.1 1719
zone remote gk-510 cisco.com 1.1.2.1 1719
zone remote gk-925 cisco.com 1.1.2.129 1719
zone remote gk-415 cisco.com 1.1.3.1 1719
zone remote gk-650 cisco.com 1.1.3.129 1719
zone prefix gk-408 408*
zone prefix gk-510 510*
zone prefix gk-925 925*
zone prefix gk-415 415*
zone prefix gk-650 650*
lrq-forwarding
. . .snipped for brevity. . .
```

As you can see, the 408 gatekeeper knows the zone prefixes of every gatekeeper in the network. In addition, the command “lrq-forwarding” allows this gatekeeper to receive Location Requests from the gatekeepers below it, and forward those LRQ’s to the appropriate destination gatekeeper.

The other gatekeepers in the network would therefore not need to know about every other gatekeeper. They could rely on the Directory Gatekeeper to resolve unknown destinations. For example, figures 5 and 6 demonstrate the configurations of the two regional gatekeepers, gk-510 and gk-415.

Figure 5 Configuration for gk-510

```
gk-510#show run
. . .snipped for brevity. . .
gatekeeper
zone local gk-510 cisco.com 1.1.2.1 1719
zone remote gk-925 cisco.com 1.1.2.129 1719
zone remote gk-408 cisco.com 1.1.1.1 1719
zone prefix gk-510 510*
zone prefix gk-925 925*
zone prefix gk-408 * (this works like a default route…)
lrq-forwarding
. . .snipped for brevity. . .
```

Figure 6 Configuration for gk-415

```
gk-415#show run
. . .snipped for brevity. . .
gatekeeper
zone local gk-415 cisco.com 1.1.3.1 1719
zone remote gk-650 cisco.com 1.1.3.129 1719
zone remote gk-408 cisco.com 1.1.1.1 1719
zone prefix gk-415 415*
zone prefix gk-650 650*
zone prefix gk-408 *
lrq-forwarding
…snipped for brevity…
```

As you can see from the above two configuration examples, these regional gatekeepers only know about their local zone prefix, and the zone prefix of the stub gatekeeper below it. All other zone prefixes are resolved by the Directory Gatekeeper (gk-408) by way of the * zone prefix. In addition, in order to receive LRQ's from their respective stub gatekeepers, and forward them to the master Directory Gatekeeper, the lrq-forwarding command is needed at the regional level as well.

Finally, let's look at how the stub gatekeepers would be configured. Figures 7 and 8 illustrate the configurations of gk-925 and gk-650 respectively.

Figure 7 Configuration for gk-925

```
gk-925#show run
. . .snipped for brevity. . .
gatekeeper
```

```
zone local gk-925 cisco.com 1.1.2.129 1719
zone remote gk-510 cisco.com 1.1.2.2.1 1719
zone prefix gk-925 925*
zone prefix gk-510 * (this works like a default route…)
. . .snipped for brevity. . .
```

Figure 8 Configuration for gk-650

```
gk-650#show run
. . .snipped for brevity. . .
gatekeeper
zone local gk-650 cisco.com 1.1.3.129 1719
zone remote gk-415 cisco.com 1.1.3.1 1719
zone prefix gk-650 650*
zone prefix gk-415 * (this works like a default route…)
. . .snipped for brevity. . .
```

As you can see, the stub-level gatekeepers only know about their own local zone prefixes, and all other zone prefixes are forwarded to their respective regional-level parent gatekeepers for resolution. Since these gatekeepers are at the lowest level of the hierarchy, the lrq-forwarding statement is not needed.

The examples above demonstrate how to use the Directory Gatekeeper feature to route zone prefixes. However, it can also be very effective for routing service prefixes as well. Lets take for example that instead of putting gateways in the various regional locations, the company has decided to place all the gateways in one data center back at the headquarters location. As their need for gateway services grow, they plan to simply add more gateways to the central rack in the 408 zone, and pool all their ISDN resources in one place, rather than distributing them out to the various locations.

In this case, rather than putting hopoff statements to “9*” in every gatekeeper in the network, the use of the “zone prefix <gk-id> *” wildcard prefix would allow every gatekeeper to forward requests upwards until they finally hit the 408 zone gatekeeper, where they would be resolved.

For example, to place a call to an H.320 terminal in Raleigh, NC (where the area code is 919), a terminal in the 650 zone would dial the following number:

9#19195551212

- When the 650 gatekeeper receives this Admission Request (ARQ) from the terminal, it would first check to see if it has any service prefixes which match the dialed number.
- It would not find any service prefixes registered in its local zone, so it would move on to try to match the dialed number against any known zone prefixes. It would not match its local zone prefix, but it would match the zone prefix of the 415 gatekeeper, which is *. Therefore, it would forward the call request via an LRQ to the 415 gatekeeper above it.
- The 415 gatekeeper would receive the LRQ from the 650 gatekeeper below it, and perform the same process. It would not find a service prefix match, nor would it match its own local zone prefix (415). However, it would match against the 408 gatekeepers zone prefix, which is *. Therefore, it would then forward the LRQ (this is where the “lrq-forwarding” command comes in to play) to the 408 zone gatekeeper above it.

- Of course, when the 408 zone gatekeeper receives the LRQ, it will immediately match dialed number to service prefiex 9#*, and reply back to the originating gatekeeper (gk-650) with the IP address and TCP signalling port information of the appropriate gateway in its zone.

This type of configuration could also be used very effectively for MCU services. If the company decided to place ALL their MCU's at one central location, it could be argued that there is no longer any need to use zone prefixes in front of the MCU service prefixes as illustrated in figure 1 *(very few customers every decide to do this, but we mention it here for purposes of our discussion of LRQ Forwarding)*. In this case, the company could choose to use service prefixes on their MCU's that are easier to dial *(such as the two-digit service prefixes that are configured on the MCU's from the factory)*. The administrator would, however, still have to ensure that the MCU service prefixes do not conflict with any zone prefixes or E.164 addresses, as described above in the section entitled, "Why service prefixes cannot conflict with any known zone prefixes or E.164 addresses".

Registering your service prefixes with the Cisco MCM Gatekeeper

This section covers the differences between the different software versions of Cisco IP/VC MCU's and Gateways, as it relates to how service prefixes are registered with the Cisco MCM Gatekeeper. It compares the configuration differences between IP/VC version 1.0, 2.0, 2.1 and 2.2 versions. This section will cover the following topics.

- How the IP/VC 3510 MCU registers its service prefixes with the Cisco MCM Gatekeeper, and how it differs between version 1.x and 2.x
- How the IP/VC 352x Gateways register their service prefixes with the Cisco MCM Gatekeeper, and it differs between versions 1.x and 2.x
- How other vendors MCU's and gateways work with the Cisco MCM Gatekeeper.

To understand the way in which MCU's and gateways register with the Cisco MCM Gatekeeper, you must first understand the difference between an E.164 address and a service prefix. If you have not yet reviewed this subject, it is covered in detail in the section above entitled, *"How the Cisco MCM Gatekeeper views service prefixes differently than it views E.164 addresses"*.

The use of E.164 addresses and service prefixes differ between H.323v1 and H.323v2 specifications. The Cisco MCM Gatekeeper conforms to H.323v2, and also is backwards compatible with H.323v1 endpoints. H.323v2 allows devices to dynamically register their service prefixes with the gatekeeper. For devices (MCU's and gateways) that are not H.323v2 compliant, you must go into the gatekeeper and statically configure the service prefixes using the "gw-type-prefix" command.

Note: *The use of the gw-type-prefix command is explained in the section above entitled,* *"When should you use the Cisco MCM Gatekeeper "hopoff" commands versus routing calls based on zone prefixes"*.

Therefore, the gateways and MCU's must be configured to register their service prefixes according to the H.323v2 specification, and not according to the H.323v1 method. This differs between versions of IP/VC software.

In version 1.x of the IP/VC software, both MCU's and gateways registered their service prefixes as E.164 addresses, NOT as service prefixes. You then had to go into the Cisco MCM Gatekeeper configuration and manually "tell" the gatekeeper that those E.164 addresses were "actually" service prefixes, and to treat them accordingly. This was done in version 1.x of the IP/VC software in order to be backwards compatible with H.323v1 gatekeepers.

In version 2.x of the IP/VC software, Cisco has changed the way in which these devices register their service prefixes. They now automatically register their service prefixes as "service prefixes",

rather than as E.164 addresses as they did before, and therefore, the use of the gw-type-prefix command is no longer needed. However, this new feature was enabled on the 3510 MCU sooner than it was on the 352x Gateways. So you must understand the differences, and which versions exhibit which type of registration behavior. Following are the details for each software release of the IP/VC product family.

How the IP/VC 3510 MCU registers its service prefixes with the Cisco MCM Gatekeeper, and how it differs between version 1.x and 2.x

In version 1.x (i.e. version 1.5 and below), the MCU registered its service prefixes as E.164 addresses, as illustrated in figure 9 below:

Figure 9

```
gk-408#show gatekeeper endpoints
                    GATEKEEPER ENDPOINT REGISTRATION
                    ================================
CallSignalAddr  Port  RASSignalAddr   Port  Zone Name         Type     F
--------------- ----- --------------- ----- ---------         ----     --
10.1.1.10       2720  10.1.1.10       2719  gk-408            H320-GW
    E164-ID: 408851
    E164-ID: 408852
    E164-ID: 408853
    E164-ID: 408854
    E164-ID: 408855
    E164-ID: 776785221075
    H323-ID: RV-MCU-221075
```

Note that since these are registered as E.164 addresses, and not as service prefixes, there is no wild-card mask (*) after the number. Therefore, this would affect the behavior of the gatekeeper as it attempts to process calls to the MCU, as explained above in the section entitled, "How the Cisco MCM Gatekeeper views service prefixes differently than it views E.164 addresses".

Thus, in order to force the gatekeeper to treat these numbers as service prefixes (i.e. logically place a * after each prefix), you had to enter the Cisco MCM Gatekeeper configuration mode, and statically define those numbers as service prefixes, using the "gw-type-prefix" command. The use of the command is as follows

gw-type-prefix *prefix#* [**hopoff** ***gkid***] [[**gw ipaddr** *ipaddr* [*port*]] ...]

For example,

```
gw-type-prefix 408851* gw ipaddr 10.1.1.10 2720
gw-type-prefix 408852* gw ipaddr 10.1.1.10 2720
gw-type-prefix 408853* gw ipaddr 10.1.1.10 2720
gw-type-prefix 408854* gw ipaddr 10.1.1.10 2720
```

```
gw-type-prefix 408855* gw ipaddr 10.1.1.10 2720
```

Note: *For more information on the use of this command, please refer to the Cisco Multimedia Conference Manager Configuration Guide.*

Note: *The Cisco IP/VC 3510 MCU uses TCP ports 2720 and 2719 for call signalling and RAS messages, rather than the well-known H.323 port numbers 1720 and 1719. The Cisco IP/VC 352x Gateways use ports 1820 and 1819.*

In Version 2.1, this registration behavior was changed. The MCU now properly registers its service prefixes with the gatekeeper under the Service Prefix Table. Therefore, the gw-type-prefix command is no longer needed. Figure 10 illustrates an MCU registration using version 2.1

Figure 10

```
gk-408#show gatekeeper endpoints
                    GATEKEEPER ENDPOINT REGISTRATION
                    ================================
CallSignalAddr  Port  RASSignalAddr   Port  Zone Name         Type    F
--------------- ----- --------------- ----- ---------         ----    --
10.1.1.10       2720  10.1.1.10       2719  gk-408            H320-GW
H323-ID: RV-MCU-221075
```

Note: *The Cisco IP/VC 3510 MCU uses TCP ports 2720 and 2719 for call signalling and RAS messages, rather than the well-known H.323 port numbers 1720 and 1719. The Cisco IP/VC 352x Gateways use ports 1820 and 1819.*

You can see that the MCU no longer registers its service prefixes under the E.164 Registration Table. Rather, they are now automatically registered under the Technology Prefix table, as shown below in figure 11.

Figure 11

```
gk-408#show gatekeeper gw-type-prefixes
GATEWAY TYPE PREFIX TABLE
=========================
Prefix: 408851*
  Zone gk-408 master gateway list:
    10.1.1.10:2720 776785221075
Prefix: 408852*
  Zone gk-408 master gateway list:
    10.1.1.10:2720 776785221075
Prefix: 408853*
  Zone gk-408 master gateway list:
    10.1.1.10:2720 776785221075
Prefix: 408854*
  Zone gk-408 master gateway list:
    10.1.1.10:2720 776785221075
```

```
Prefix: 408855*
  Zone gk-408 master gateway list:
    10.1.1.10:2720 776785221075
```

This is the expected behavior of all future releases of MCU software, such as 2.2 and beyond.

Note: *You may notice that the MCU registers as a DeviceType of "H320-GW". This is because the Cisco MCM Gatekeeper does not currently support the notion of service prefixes for MCU's. This is expected to change in the future. For the time being, MCU's must register themselves as "gateways" in order to offer support for service prefixes. If a device registers as an MCU it is allowed to register E.164 addresses, but it may not register service prefixes with the gatekeeper. This affects interoperability with other vendors MCU's. We will deal with this in more detail in the section below entitled, "How other vendor's MCU's and gateways work with the Cisco MCM Gatekeeper".*

How the IP/VC 352x Gateways register their service prefixes with the Cisco MCM Gatekeeper, and how it differs between versions 1.x and 2.x

Similarly to the MCU version 1.x, the Cisco IP/VC 352x Gateways register their service prefixes as E.164 addresses. ***Unlike the MCU, which was modified as of version 2.x, the Gateways continue to do this in version 2.0 as well***.

However, in version 2.2 and beyond, the Gateways are expected to register their service prefixes properly, as in MCU version 2.1, described above. Figure 12 illustrates a gateway registration, using both version 1.x and version 2.0.

Figure 12

```
gk-408#show gatekeeper endpoints
                    GATEKEEPER ENDPOINT REGISTRATION
                    ================================
CallSignalAddr  Port  RASSignalAddr   Port  Zone Name         Type    F
--------------- ----- --------------- ----- ---------         ----    --
10.1.2.20       1820  10.1.2.20       1024  gk-510            H320-GW
    E164-ID: 9#
    E164-ID: 90#
    E164-ID: 91#
    E164-ID: 92#
    H323-ID: GW3675735979
```

Note that since these are registered as E.164 addresses, and not as service prefixes, there is no wild-card mask (*) after the number. Therefore, this would affect the behavior of the gatekeeper as it attempts to process calls to the gateway, as explained above in the section entitled, "How the Cisco MCM Gatekeeper views service prefixes differently than it views E.164 addresses".

Thus, in order to force the gatekeeper to treat these numbers as service prefixes (i.e. logically place a * after each prefix), you have to enter the Cisco MCM Gatekeeper configuration mode, and statically define those numbers as service prefixes, using the "gw-type-prefix" command. The use of the command is as follows

gw-type-prefix *prefix#* [**hopoff** *gkid*] [[**gw ipaddr** *ipaddr* [*port*]] …]

For example,

```
gw-type-prefix 9#* gw ipaddr 10.1.2.20 1820
gw-type-prefix 90#* gw ipaddr 10.1.2.20 1820
gw-type-prefix 91#* gw ipaddr 10.1.2.20 1820
gw-type-prefix 92#* gw ipaddr 10.1.2.20 1820
gw-type-prefix 93#* gw ipaddr 10.1.2.20 1820
```

Note: *For more information on the use of this command, please refer to the Cisco Multimedia Conference Manager Configuration Guide.*

Note: *The Cisco IP/VC 352x Gateways use TCP ports 1820 and 1819 for call signalling and RAS messages, rather than the well-known H.323 port numbers 1720 and 1719. The Cisco IP/VC 3510 MCU uses ports 2720 and 2719.*

In version 2.2 and beyond, this behavior is expected to change, and the gateways will register their service prefixes properly, as in MCU version 2.1, described above.

How other vendor's MCU's and gateways work with the Cisco MCM Gatekeeper.

Finally, we will discuss how the Cisco MCM Gatekeeper views MCU's, and how that effects interoperability with other vendor's MCU products. As noted above in the MCU examples given in figures x and x, the Cisco IP/VC 3510 MCU is configured from the factory to register as an "H320-GW", rather than as an "MCU". This is because the H.323 specification clearly specifies the use of service prefixes on gateways, but it is not clearly defined for MCU's. It is just as feasible for MCU's to use service prefixes, but because it was not clearly defined in the standard, this has become a "gray area" between vendors, and is an interoperability issue.

RADVision, who championed the way for H.323 MCU's, implemented the use of service prefixes, and other vendors (e.g. Accord Networks) have followed their lead. However, the Cisco MCM Gatekeeper does not allow devices that register as an "MCU" to use service prefixes. It only allows gateways to do so.

As a result, in order to enable the use of service prefixes on our MCU's, Cisco modifies the 3510 MCU to cause it to register as a gateway. Other vendor's MCU's do not allow that parameter to be modified. For example, as of this month - December 2000 - Accord Network's MGC-50 and MGC-100 products will only register as an MCU.

This problem is expected to be resolved in future versions of the products. However, it is too early to predict what form this resolution will take. There are several options available. For one, Cisco could modify the MCM Gatekeeper. Alternatively, other MCU vendors could enable a configuration parameter to modify their registration mode. Both options are valid, until the standard is more clearly defined.

Note: *Until it is properly fixed, there are a couple of work-arounds available for this problem. If you have another vendor's MCU product, and want it to work with the Cisco MCM Gatekeeper, please contact cs-ipvc@cisco.com, or ipvc-support@external.cisco.com for additional assistance.*